DEVS MODELS FOR ARCTIC MAJOR MARITIME DISASTERS

Hazel Griffith[1], and Gabriel A. Wainer[1]

[1]Dept. of Systems and Computer Eng., Carleton University, Ottawa, ON, CANADA

ABSTRACT

Modern modelling and simulation techniques allow us to safely test the policies used to mitigate disasters. We show how the DEVS formalism can be used to ease the modelling process by exploiting its modularity. We show how a policymaker's existing models of any type can be recreated with DEVS so they may be reused in any new models, decreasing the number of new models that need to be made. We recreate a sequential decision model of an arctic major maritime disaster developed by the Canadian government as a DEVS model to demonstrate the method. The case study shows how DEVS allows policymakers to create models for studying emergency policies with greater ease. This work shows a method that can be used by policymakers, including models of emergency scenarios, and how they can benefit from creating equivalent DEVS models, as well as exploiting the beneficial properties of the DEVS formalism.

1   INTRODUCTION

When a dangerous situation occurs, like a flood, a forest fire, a tornado, or any major evacuation event, difficult choices need to be made quickly to save the largest number of people. In these situations, there are often limited resources. Therefore, to prepare for these dangerous situations it is best to have policies on what to do with the resources that are immediately available. In such dangerous scenarios we rely on these policies to be an effective means to keep the largest number of people safe, but there is usually little time to compare the merits of different policies when an emergency occurs. Therefore, it is necessary to conduct experiments with different policies before an emergency occurs to determine whether a new policy is worth using instead of a currently accepted policy.

Modelling and simulation (M&S) techniques can be used to recreate such emergencies in a safe environment, as if we tried to conduct experiments with real emergency situations, there could be a risk of danger to the people involved and complex ethical consideration. Instead, if we build models that represent the aspects of interest in the emergency and then execute simulations with different policies, we can address important aspects of an emergency and enact policies when the actual emergency occurs. Nonetheless, modeling and simulation techniques often incur substantial development costs. The definition of an emergency model is a process that can be complex, and when we study situations that result in a loss of human life, every decision must be extensively justified. This critical verification and validation leads to longer development times, hiring expert consultants, and expensive software licenses.

Although model reuse can help to reduce these costs, most models are built for one specific set of circumstances and changing them is a laborious task because of the formalism they were designed with. However, there are various modelling techniques that allow for this kind of modularity. One of them, called DEVS (Discrete-Event system Specifications) is a M&S formalism for defining discrete-event models (Zeigler et al. 2019). With the DEVS formalism a real system can be broken down into a combination of smaller models that represent specific aspects of the system. If our system changes, it is easy then to replace one or more models to update the system's model. Thanks to DEVS modularity we can create a repository of models that can be pulled from whenever a model of a new system needs to be defined.

We show how to benefit from using DEVS when modelling emergency situations to solve the problems discussed above. We demonstrate this with a case study on using DEVS to meet the Canadian government's

need for models to study different major maritime evacuations in the arctic north. We take one of their most recently created models for studying evacuation policies used in a ship evacuation from the Northwest Passage and recreate its components as DEVS models that can be reused and combined in different ways to quickly create models of different arctic maritime evacuation scenarios. This allows creating a repository of DEVS models to be reused, instead of creating new models for every different scenario.

2   RELATED WORK

Simulation models have been developed for managing many different potential disaster scenarios, using discrete-event, agent-based, and Monte-Carlo simulations. Discrete-event modelling techniques allow you to abstract a complex situation into discrete states that are easier to represent. For example, Kwok et al. (2019) use discrete-event modelling to develop a model of a metro train evacuation, showing how a disaster would change over time. Similarly, Debacker et al. (2016) present a mass casualty incident where a disaster has occurred, and many people require medical care. The paper by Basaglia et al. (2022) uses discrete-event simulation to model patient flow through a hospital after an earthquake has inundated the hospital with injured people.

Agent-based modelling techniques are useful for modelling disaster scenarios, as it allows researchers to analyze emergent behaviour and unexpected outcomes amongst the acting bodies involved in a crisis. For example, in the paper Fikar et al. (2018), the researchers used an agent-based simulation to test the effectiveness of dynamic disaster relief distribution solutions generated by an optimization model. The agent-based model consisted of many agent models that represented individual people, vehicles, and locations. This includes what tasks the agents perform and how they interact with each other. Another paper (Schoenharl & Madey, 2011) uses an agent-based model of a group of people's mobile phone activity to simulate the differences in user location and mobile phone usage when a crisis has and has not occurred. This is useful for training classification and prediction models to recognize when a crisis event has happened. Agent-based modeling is also used in the paper Wagner & Agrawal (2014) to model the evacuation of a concert venue with the presence of a fire. The people trying to evacuate and the spreading fires are represented by agent-based models to mimic their real behaviour.

Monte Carlo models supply the inputs of a deterministic system with random values according to the probability distributions of each input. The outcome for each set of random inputs is recorded and compared against other outcomes to determine the likelihood of an event occurring with the system being tested. For instance, Banomyong & Sopadang (2010), present a Monte Carlo model for developing emergency logistics response, applied to a case study of a tsunami off the coast of Thailand. In Lumbroso & Davison (2018), the Monte Carlo model is used to analyze the results of an agent-based model that depicts the outcome of flooding on Canvey Island in the Thames Estuary. These methods are useful for comparing different emergency management procedures.

Different research studies the Canadian Armed Forces' (CAF) response to a major maritime (MAJMAR) disaster scenario in the northern arctic. The scenarios consider a cruise ship carrying two thousand people through the northwest passage that needs to be evacuated because of an accident. People are evacuated from the cruise ship by lifeboats to the nearest shore which becomes the evacuation site. Unfortunately, there are no possible evacuation sites where large aircraft can land, so a forward operating location (FOL) within helicopter range of the evacuation site is established. The evacuees will be categorized by triage level and then prioritized for evacuation by their level.

The papers Hunter et al. (2021) and Hunter et al. (2019) use mixed-integer programming to study the sensitivity of existing infrastructure in the Arctic to investment or divestment decisions given its ability to aid in a MAJMAR disaster. Their models both consider how the evacuees health will deteriorate over time, how sufficient fuel needs to be made available for helicopters, how a FOL within helicopter range of the evacuation site needs to be established, how resources will be transported to the FOL given the current CAF deployment and response posture, and the vehicle capacities.

Similarly, Rempel et al. (2021), Rempel & Shiell (2022), and Rempel (2024) focus on the process of evacuating individuals from the evacuation site to the FOL. They use a Markov Decision Process (MDP)

to describe the process of prioritizing evacuees by triage level to be loaded onto vehicles. Approximate dynamic programming was used to generate Value Function Approximation (VFA) based prioritization policies. Then, the research used Powell's universal framework for sequential decision problems instead of a MDP to model a scenario where the coast guard ship stays at the evacuation site to provide medical aid while only helicopters are used for the evacuation, finding a green-first policy is best for loading the ship, and a critical-first policy is best for loading the helicopter.

There are many methods that can be used to study evacuations, but DEVS has been shown to be a suitable choice. For example, Wang et al. (2012), used DEVS to automatically construct evacuation models from building information modelling files. In Zhang et al. (2014), the researchers demonstrated how DEVS can be used to create a transportation evacuation model that can execute agent-based simulations faster than a time-step based model. Bae et al. (2014) modeled the evacuation of a city under siege, combining an agent-based model with DEVS to accurately model the military force bombarding the city and the evacuating civilians. Ha et al. (2012) used Cell-DEVS to model the passengers of a ship evacuating to lifeboats.

3 ARCTIC MAJOR MARITIME DISASTER SCENARIO

A major maritime disaster (MAJMAR) is a scenario where a large number of people are in danger at sea. In particular, the Canadian government is interested in MAJMAR scenarios involving cruise ships passing through the arctic Northwest passage. This is becoming more common as climate change progresses (Weber, 2016). The population in this region is sparse, so there is little infrastructure that could be used to aid an evacuation. The Canadian Coast Guard (CCG) would normally handle the evacuation, but they do not have enough resources, so the Canadian Armed Forces (CAF) would take control. With the CAF's large planes, they could evacuate people quickly, but of the possible evacuation sites along the Northwest passage none are close to a runway for planes to land. As such, the CAF would need to establish a Forward Operating Location (FOL) within helicopter range of the evacuation site. The evacuees could be brought by helicopter to the FOL and then transported by plane to a safe location like Trenton, Ontario. However, the helicopters available to the CAF have a much smaller capacity. To decide who is evacuated first the CAF would follow a policy that prioritizes evacuees by their assigned triage level. The Simple Triage And Rapid Treatment method (START) assigns a color to victims that indicates how urgently they require medical care. White means the person does not require treatment, green means they need routine treatment, yellow indicates they need treatment soon, red means they need treatment immediately, and black indicates the person is deceased. A priority policy orders these colors according to which passenger gets evacuated first. A critical-first policy would use the order, "Red, Yellow, Green, White". Black is not included as they cannot be rescued. In this scenario, the CCG is able to dock their ship at the evacuation site and provide medical aid to a relatively small number of evacuees at a time. Inside the CCG ship it is assumed that an evacuee's medical status can only improve, but outside the CCG ship at the evacuation site it is assumed that as they are exposed to the harsh arctic environment an evacuee's medical status can only deteriorate. This change in medical status is represented by an evacuee's triage level moving to the next better or worse level. To determine who gets medical aid from the CCG ship first another priority policy would be used. Once every 24 hours, a third priority policy would determine who is to be removed from the CCG ship so that others may receive aid.

With this scenario into consideration, Rempel (2024) modeled the arctic MAJMAR scenario using Powell's universal modelling framework for sequential decisions under uncertainty. This method represents a situation as a series of sequential decisions (Powell 2022). In this case, the decisions are which people are evacuated by helicopter and loaded/unloaded from the CCG ship next. The uncertainty comes from randomly sampling the times until an evacuee's health changes from an exponential distribution whose mean is dependent on their current medical status, as someone in the red level would deteriorate faster than they improve. The sequential decision model is defined by *state variables* that describe the current state of the model, *decision variables* that are the choice being made to control the situation, *exogenous information variables* which influence the state outside of our control, a *transition function* that calculates what the next

state will be given the current state and the exogenous information received, and an *objective function* used to determine the quality of the choices made. The goal of a sequential decision model under uncertainty is to maximize he expected value of the objective function by adjusting the decisions made. In (Rempel, 2024) the model's *state variables* are the time step, the event code, the number of evacuees in each triage level outside at the evacuation site, and the number of evacuees in each triage level on board the CCG ship. The event code determines what decision needs to be made. Depending on whether the ship needs to be loaded/unloaded or the helicopter needs to be loaded, the *decision variable* is the number of evacuees in each level being moved. This decision is made according to the priority policy currently being tested. The *exogenous information variables* are the evacuees whose medical status changed. The *objective function* used is the sum of all evacuees who were evacuated by helicopter. Different combinations of priority policies were tested by executing the model and comparing the expected value of the contribution function. Thirty two different combinations of priority policies were tested.

4   BUILDING A MAJMAR DISASTER SCENARIO IN DEVS

As discussed in the Introduction, the DEVS formalism for modeling has some advantages over other modeling techniques that could be applied to the model presented in Rempel (2024). DEVS could be used to study the MAJMAR scenario by defining a combination of four major components: the evacuees, the helicopters, the evacuation site, and the FOL. We discuss how such models were built using DEVS to represent each component and implemented with the Cadmium library. The top model shown in (Figure 1) is the DEVS coupled model we designed for the arctic MAJMAR scenario. It consists of three coupled models and one atomic model. These models represent the evacuation site, helicopters, evacuees, and the FOL. The connecting lines between them illustrate how messages are exchanged between the models. Multiple *Evacuee* and *Helicopter* models are instantiated to represent each evacuee and helicopter in a given experiment. Such an experiment would execute until no more evacuees outside of the black triage level remain at the evacuation site.
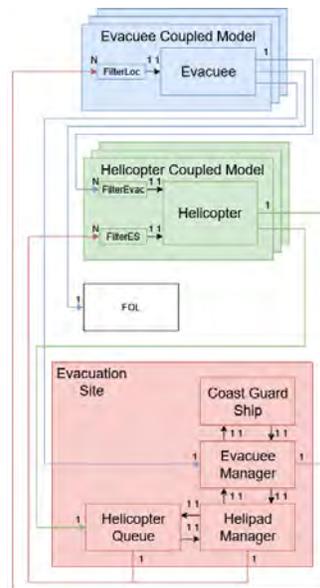

Figure 1: DEVS top model of MAJMAR scenario.

The model in Figure 2 keeps track of the evacuee's current location, their triage level, and how long it will be until their triage level changes. If the evacuee's current location is the evacuation site and outside the CCG ship, their health is deteriorating. The time until their triage level moves to the next worse color code is randomly sampled from an exponential distribution whose mean is dependent on what triage level

the evacuee is currently in Rempel (2024). This has been used in for priority assignment in emergency response scenarios (Jacobson et al. 2012).
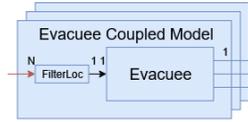

Figure 2: DEVS evacuee coupled model.

Figure 3 shows the *Evacuee* atomic model and all the related functions in the MAJMAR scenario. Each circle represents a state, including the time advance for recording the time until advancing to the next state, and the current message received by the model. Arrows with a dashed line indicate internal transitions while arrows with a solid line indicate external transitions. The evacuee's current location where ES refers to the evacuation site, CGS refers to the coast guard ship, and FOL stands for the forward operating location. The state variable TS stands for triage status, and it indicates the evacuee's current health status as either W for white, G for green, Y for yellow, R for red, and B for black. The variable *heloID* represents the helicopter that the evacuee has been assigned; it is -1 when a helicopter has not been assigned to them yet. Lastly, the Time Advance (TA) variable denotes the amount of time left until the next internal transition. For some states this is set to *ExpDist* which represents randomly sampling a value from an exponential distribution with the given mean. If TA is set to *inf* then it will remain in that state until an external transition occurs.
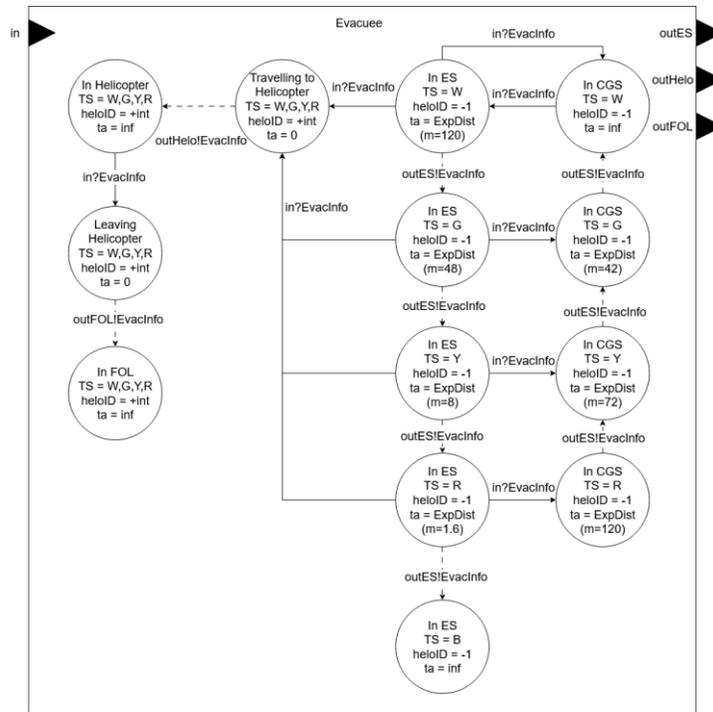

Figure 3: Evacuee atomic model DEVS graph.

Similarly, *FilterLoc* is used to ensure that only messages with a matching evacuee *ID* are passed on to the connected Evacuee model. This is necessary as there could be any number of *Evacuee* models. The DEVS graph in (Figure 4) shows all possible behaviors of *FilterLoc*.

(Figure 5) shows the *Evacuation Site* model, which models how evacuees are selected for evacuation and medical treatment, how incoming helicopters queue up for the one helipad, and when the coast guard ship arrives and loads/unloads evacuees. The incoming blue arrow indicates that an input is being received

from the *Evacuee* model, and it is sent to the *Evacuee Manager* when an evacuee's triage level has changed. The model sends outputs from *Helicopter Queue* to the *Helicopter* to notify helicopters that they can land; the *Helipad Manager* sends outputs to *Helicopter* to tell the current *Helicopter* that it can take off, and the *Evacuee Manager* sends the *ID* of the helicopter to the *Evacuee* that has been assigned to for the evacuation.
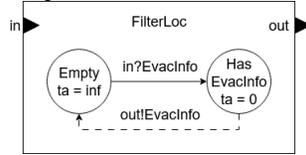

Figure 4: FilterLoc atomic model DEVS graph.

Helicopter Queue accepts helicopter *IDs* from helicopters that are looking to land at the evacuation site. When the Helipad Manager has a vacant helipad it sends a message to the *Helicopter Queue* that the next helicopter can land. *Helipad Manager* manages the use of the helipad to ensure that only one helicopter can land at a time. Once the *Helipad Manager* receives a message back from *Helicopter Queue* containing a helicopter's *ID,* it sets that helicopter as the one which is currently landed. *Helipad Manager* will then wait for 15 minutes before sending a message to *Helicopter* with the associated *ID* to indicate that it is time to take off. At the same time, it sends the helicopter *ID* to the *Evacuee Manager* to indicate that this helicopter must be sent the selected evacuees. *Evacuee Manager* decides where evacuees are located, organizing them by their triage level. When it receives a message from the *Evacuee* (with the updated triage level of an *Evacuee),* it updates its triage level queue. When it receives a message from the *Coast Guard Ship,* it selects evacuees that are not on board the *Coast Guard Ship* from the triage level queues according to the ship loading policy and sends a message to each *Evacuee* with their new location on board the ship for medical treatment. The *Coast Guard Ship* represents the CCG ship sent to the evacuation site to provide medical treatment for the evacuees. It tracks when the CCG ship is meant to arrive, which can be between zero and one hundred and sixty-eight hours, depending on the experiment.
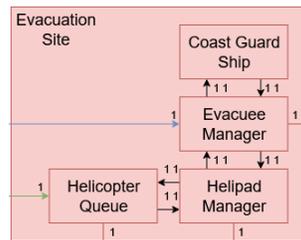

Figure 5: *Evacuation Site* DEVS coupled model.

Figure 6 shows a DEVS model representing a single helicopter used to transport evacuees from the evacuation site to the FOL. It keeps track of the helicopter's current location, how long it takes to move to a different location, and which evacuees are on board the helicopter. *FilterEvac* and *FilterES* filter messages received by all *Helicopter* coupled models such that each *Helicopter* atomic model only receives the messages they are intended to receive. *FilterEvac* receives messages from evacuees with their triage level and identification number. There is also an atomic model to represent the FOL, which keeps track of the evacuees that have been evacuated from the evacuation site. This makes taking statistics around the number of surviving evacuees easier to acquire. The *FOL* atomic model receives messages from *Evacuees* that have been transported to the FOL by a helicopter. These messages include their *Evacuee ID* and triage level.
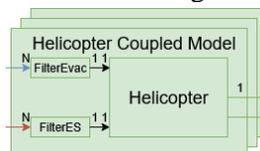
Figure 6: Helicopter DEVS coupled model.

The models were implemented in C++ with the Cadmium V2 library (Cardenas et al. 2023). The Evacuee atomic is responsible for tracking when an evacuee's health changes enough to improve or deteriorate to the next triage level. This is handled by setting the *time advance* state variable to the amount of time left until their health status changes. This time is randomly sampled from an exponential distribution with a mean dependent upon their current location and triage level. If an evacuee is in the green triage level and on board the ship their health status will improve after an amount of time taken from an exponential distribution with a mean of forty-eight hours. When the internal transition executes, the evacuee's triage level either improves if they are on board the ship or deteriorates if they are not. If an external transition occurs saying that the *Evacuee* is either boarding or leaving the ship, then the triage level should not be changed but the time until their health status changes must be resampled from a different distribution. A part of the definition of the *Evacuee's* internal transition function can be seen below.

```
if (state.curr_loc == 'E'){
        unsigned seed1 = chrono::system_clock::now().time_since_epoch().count();
        minstd_rand0 generator(seed1);
        exponential_distribution<float> wTogDistribution{float(1.0/m_wTog)};
        exponential_distribution<float> gToyDistribution{float(1.0/m_gToy)};
        exponential_distribution<float> yTorDistribution{float(1.0/m_yTor)};
        exponential_distribution<float> rTobDistribution{float(1.0/m_rTob)};
        if (state.start){
                state.start = false;
                switch(state.triage_status){
                        case('W'):
                                state.sigma = (double) wTogDistribution(generator);
                        // ... The process repeats for the remaining triage categories
                        }
        } else {
                switch(state.triage_status){
                        case('W'):
                                state.triage_status = 'G';
                                state.sigma = (double) gToyDistribution(generator);
                        // ... The process repeats for the remaining triage categories
} else if (state.curr_loc == 'C'){
        unsigned seed1 = chrono::system_clock::now().time_since_epoch().count();
        minstd_rand0 generator(seed1);
        exponential_distribution<float> rToyDistribution{float(1.0/m_rToy)};
        exponential_distribution<float> yTogDistribution{float(1.0/m_yTog)};
        exponential_distribution<float> gTowDistribution{float(1.0/m_gTow)};
        if (state.firstTimeOnShip){
                state.firstTimeOnShip = false;
                switch(state.triage_status){
                        case('W'):
                                state.sigma = numeric_limits<double>::infinity();
                        // ... The process repeats for the remaining triage categories
        } else {
                switch(state.triage_status){
                        case('W'):
                                state.sigma = numeric_limits<double>::infinity();
                // ... The process repeats for the remaining triage categories
```

Figure 9: *Evacuee* model code for managing the triage level.

The coupled top model contains three submodels, so each has its own class from which coupled models are generated by combining the correct atomic models. The *Evacuee* coupled model accepts a string as its model *id,* a unique positive integer as its *evacueeID,* and a character to initialize its triage level. In the *Evacuee* coupled model one *Evacuee* atomic model and one *filterLoc* atomic model are initialized with their necessary ports and connected. The code for the *Evacuee* coupled model constructor can be seen below.

```
EvacueeCoupled(const std::string& id, int evacueeID, char triage_status): Coupled(id){
shared_ptr<Evacuee> evac = addComponent<Evacuee>("evacuee", evacueeID, triage_status);
shared_ptr<FilterLoc> filterLoc = addComponent<FilterLoc>("filterLoc", evacueeID);
in = addInPort<EvacInfo>("in");
        outHelo = addOutPort<EvacInfo>("outHelo");
        outFOL = addOutPort<EvacInfo>("outFOL");
        outES = addOutPort<EvacInfo>("outES");
        addIC("filterLoc", "out", "evacuee", "in");
        addEIC("in", "filterLoc", "in");
        addEOC("evacuee", "outHelo", "outHelo");
        addEOC("evacuee", "outFOL", "outFOL");
        addEOC("evacuee", "outES", "outES");
}
```

Figure 10: *Evacuee* coupled model constructor.

Similarly, the *EvacuationSiteCoupled* class creates the *Evacuation Site* coupled model. The constructor accepts the model's *id*, *timeToLoad* that equals the time required to load a helicopter, a list of initialized evacuees, and *startTime,* which is equal to the time when the ship arrives at the *Evacuation Site*. The constructor then creates the *HelicopterQueue, HelipadManager, EvacueeManager,* and *CoastGuardShip* models. Their ports are created and connected by them, as seen in the code below.

```
EvacuationSiteCoupled (const std::string& id, double timeToLoad, vector<EvacInfo> evacuees,
              double startTime): Coupled(id){
   shared_ptr<HelicopterQueue> hq = addComponent<HelicopterQueue>("helicopterQueue");
   shared_ptr<HelipadManager> hm = addComponent<HelipadManager>("helipadManager", timeToLoad);
   shared_ptr<EvacueeManager> em = addComponent<EvacueeManager>("evacueeManager", evacuees);
   shared_ptr<CoastGuardShip> cgs = addComponent<CoastGuardShip>("coastGuardShip", startTime);
   inEvac = addInPort<EvacInfo>("inEvac");
   inHelo = addInPort<HeloInfo>("inHelo");
   outHelo = addOutPort<HeloInfo>("outHelo");
   outEvac = addOutPort<EvacInfo>("outEvac");
   addIC("helicopterQueue", "outHM", "helipadManager", "inHQ");
   addIC("helipadManager", "outHQ", "helicopterQueue", "inHM");
...
   addIC("coastGuardShip", "out", "evacueeManager", "inCGS");
   addEIC("inEvac", "evacueeManager", "inEvac");
   addEIC("inHelo", "helicopterQueue", "inHelo");
   addEOC("evacueeManager", "outEvac", "outEvac");
   addEOC("helicopterQueue", "outHelo", "outHelo");
   addEOC("helipadManager", "outHelo", "outHelo");
}
```

Figure 12: *EvacuationSiteCoupled* constructor.

The top model is also implemented as a class in C++. It initializes the inputs for each coupled model's class constructor, creates all three coupled models along with the *FOL* atomic model, and connects them by their ports. To run experiments the *Top_model* class's constructor accepts a double called *shipArrivalTime* that is the time in minutes until the CCG ship arrives, and a positive integer called *numOfHelos* that is the number of helicopters in the experiment. This allows us to run different kinds of experiments without rebuilding the C++ executable as the parameters of the experiment can be input as command line arguments. These command line arguments are handled by the main function in the *main.cpp* file where the top model, Cadmium root coordinator, and simulation logger are instantiated.

All the atomic models have been defined as discussed above; they will not be included due to space constraints. The interested reader can find them in the MAJMAR_DEVS github (Griffith 2024).

5   VALIDATION OF DEVS MAJMAR SCENARIO EXPERIMENTS

The DEVS model of the MAJMAR scenario successfully passed all the verification tests, but those only ensured that it functions according to the model specifications. To determine whether the model specifications depict reality correctly, they need to be validated. As this is a DEVS version of an existing model that has been used for many different experiments, and validated by domain experts, we run the same experiments and checked whether we get results that are significantly different or not.

We first conducted an experiment that focuses on measuring the number of lives saved if the loading and unloading policies are fixed, but the number of helicopters used changes and the arrival time of the CCG ship varies (Rempel 2024). This can be used for decision support in similar scenarios. The   GF policy is used for loading both the helicopter and the CCG ship, and the WTO unloading policy is used for unloading the CCG ship. GF was chosen as the loading policy because it showed promising results in earlier research (Rempel et al. 2021; Rempel and Shiell 2022), where they attempted to generate the best possible loading policy. The WTO policy is chosen, as ship unloading policies have not been explored by earlier research, so a sympathetic policy is best until more research has been conducted. Three types of scenarios with one, three, and six helicopters evacuating people to the FOL from the evacuation site were chosen. The CCG ship arrival times start at zero hours since the evacuees arrived at the evacuation site, and one hundred sixty-eight hours after the evacuees arrived. These were chosen as they are consistent with the expected resources available and initial conditions in such a situation.

The *FOL* atomic model was used to store the evacuees that have been successfully transported from the evacuation site, so once a simulation has finished running, we can check the logs generated by the *FOL* atomic model to find the number of people saved. However, the exponential distributions introduce a stochastic element to the model, such that we cannot assume that the number of lives saved from one execution of an experiment will always be the same. To get an expected number of lives saved for each experiment we run each experiment as many times as to get a confidence interval of around one evacuee. As a result every experiment was run with 100 replications.

As we can see in Table 1, there were no significant differences between the results obtained from the experiment conducted with the DEVS model and with the results obtained from the same experiment with the sequential decision model. We used these results to consider that the DEVS model's specifications are valid, as the sequential decision model's specifications were validated earlier, it was confirmed that the results obtained were valid.

Table 1: Results of experiment where the ship arrival time and the number of helicopters varies.

Ship Arrival Time (h)	1 Helicopter		3 Helicopters		6 Helicopters	
	Mean (Lives saved)	Variance	Mean (Lives Saved)	Variance	Mean (Lives saved)	Variance
0	180.35	0.1892	207.44	0.2672	218.42	0.3074

12	175.08	0.2153	200.65	0.3469	210.1	0.2411
24	165.75	0.1702	192.54	0.2316	201.59	0.3231
36	158.4	0.1189	183.8	0.2873	193.98	0.1998
48	149.83	0.1616	177.39	0.2147	187.73	0.1730
60	142.27	0.1447	171.44	0.2199	179.88	0.2363
72	136.87	0.1398	165.32	0.2172	180.35	0.1601
84	126.16	0.1347	164.41	0.1871	179.41	0.2462
96	125.38	0.1088	164.49	0.1475	179.18	0.1822
108	125.73	0.1472	163.59	0.2103	179.05	0.2764
120	125.66	0.1543	165.37	0.1685	179.17	0.1987
132	125.47	0.1292	163.23	0.1856	179.23	0.2025
144	125.31	0.1476	165.54	0.1545	179.94	0.2266
156	125.53	0.1404	165.59	0.1461	179.86	0.1824
168	126.36	0.1241	163.26	0.1569	179.72	0.2334
Never	125.62	0.1272	164.48	0.1702	180.11	0.1938

As the number of helicopters increases, we see a diminishing return on the number of lives saved. This is because the evacuation site will likely only have one suitable landing pad for the helicopters, requiring helicopters to wait for access to the landing pad. This leads to a smaller increase in the percentage of evacuees being transported to the FOL. The helicopters also take the same amount of time to reach the evacuation site, so regardless of the number of helicopters used, many evacuees in the yellow and red triage categories will perish before they can be loaded into a helicopter.

The experiment results show that the number of lives saved converges to the same value as the ship's arrival time increases. Indicating that if the ship arrives too late it will have no impact on the number of lives saved. This ship arrival time where the number of lives saved converges, follows a clear trend as it decreases as the number of helicopters increases. For one helicopter the number of lives saved converges at the eighty-four-hour ship arrival time, for three helicopters the convergence occurs at the seventy-two hour mark, and for six helicopters it is at the sixty hour mark. The ship arrival time where the convergence occurs is likely decreasing because more helicopters make it slightly faster to evacuate everyone, so the ship needs to arrive sooner to have an impact. The convergence time decrease appears to be proportional to the difference in the number of lives saved with different numbers of helicopters, but more experiments with smaller differences in the ship arrival times are required to confirm this.

6   CONCLUSION

This paper showed how to benefit from using the DEVS formalism when modelling emergency situations to study the policies that might be used. We explored the use of discrete-event system specifications in adapting policymakers' existing models to build models of new emergency situations at a reduced cost. This is accomplished by following a four-step process. First, we take the emergency situation that a model represents and break it down into the independent parts we want to study. Second, we develop DEVS models of those parts. Third, we connect those DEVS models to form a model of the entire emergency situation. And fourth, we conduct the same experiments on the DEVS model that we have results for with the original model and compare the results to validate the DEVS model. In the case study presented, we used a sequential decision model developed by the Canadian government to study major maritime disasters in the Northwest Passage. We divided the scenario into DEVS models of each part and coupled them together to reform the major maritime disaster depicted by the sequential decision model. We conducted an experiment that matched an experiment conducted on the sequential decision model, showing that our method produces correct results. This would allow the researchers who are studying major maritime disasters to improve the quality of their models when creating new but similar models by using any or all the DEVS models we developed. This is the case as the DEVS modelling and simulation formalism has the modular and hierarchical properties needed to easily support reusing existing models in new contexts and

to easily support the parallel execution of models in the same simulation. Policymakers with an existing repository of models could follow the steps we outlined to create equivalent DEVS models that benefit from the properties we discussed. However, there are some limitations. For a DEVS model to be reused in a different context it might need to be parameterized such that the DEVS model can be changed to fit the new context. For example, the *Evacuee* DEVS model we created has their health deteriorate at a rate randomly sampled from an exponential distribution with a fixed mean. If another researcher wanted to change how the rates of health deterioration are calculated they would need to change the source code of the *Evacuee* model. It would be trivial to parameterize the *Evacuee* model where a researcher could enter their own method for determining the rate at which an evacuee's health changes. But the trick is in anticipating that that is something a researcher might want to change. So when creating a DEVS model to be reused the designer must decide what can and cannot be parameterized without losing the validity of the model. As in the context of the Canadian research into MAJMAR scenarios, it must use an exponential distribution with those fixed means for its results to be valid.

REFERENCES

Adini, B., Cohen, R., Glassberg, E., Azaria, B., Simon, D., Stein, M *et al*. 2014. "Reconsidering Policy of Casualty Evacuation in a Remote Mass-Casualty Incident". *Prehospital and Disaster Medicine* 29(1):91-95.

Badiali, S., Giugni, A., and Marcis, L. 2017. "Testing the START Triage Protocol: Can It Improve the Ability of Nonmedical Personnel to Better Triage Patients During Disasters and Mass Casualties Incidents?". *Disaster Medicine and Public Health Preparedness* 11(3):305-309.

Bae, J. W., Lee, S., Hong, J. H., and Moon, I.-C. 2014. "Simulation-based analyses of an evacuation from a metropolis during a bombardment". *SIMULATION* 90(11):1244-1267.

Banomyong, R., and Sopadang, A. 2010. "Using Monte Carlo simulation to refine emergency logistics response models: a case study". *International Journal of Physical Distribution & Logistics Management* 40(8/9):709-721.

Barry, C., and Zain, A. 2018. "Development of an evacuation policy at a healthcare facility". *The Southwest Respiratory and Critical Care Chronicles,* 6(26):41-45.

Basaglia, A., Spacone, E., Van de Lindt, J. W., and Kirsh, T. D. 2022. "A Discrete-Event Simulation Model of Hospital Patient Flow Following Major Earthquakes". *International J of Disaster Risk Reduction* 71:102825.

Cardenas, R., Trabes, G., Soulier, J., Menard, J., 2023. "cadmium_v2". https://github.com/SimulationEverywhere/cadmium_v2, accessed 29th April 2025.

Debacker, M., Van Utterbeeck, F., Ullrich, C., Dhondt, E., and Hubloue, I. 2016. "SIMEDIS: a Discrete-Event Simulation Model for Testing Responses to Mass Casualty Incidents". *Journal of Medical Systems,* 40(273).

Fikar, C., Hirsch, P., and Nolz, P. C. 2018. "Agent-based simulation optimization for dynamic disaster relief distribution". *Central European Journal of Operations Research,* 26:423-442.

Gebhart, M. E., and Pence, R. 2007. "START Triage: Does It Work?". *Disaster Management & Response,* 5(3), 68-73.

Griffith, H. 2024. "MAJMAR_DEVS". https://github.com/HazelGriffith/MAJMAR_DEVS accessed 3rd March 2025.

Ha, S., Ku, N.-K., Roh, M.-I., and Lee, K.-Y. 2012. "Cell-based evacuation simulation considering human behavior in a passenger ship". *Ocean Engineering* 53:138-152.

Hunter, G., Chan, J., and Rempel, M. 2019. "Assessing the Impact of Infrastructure on Arctic Operations". [STO-MP-SAS-OCS-ORA-2019], NATO S&T Organization.

Hunter, G., Chan, J., and Rempel, M. 2021. "Assessing the Impact of Infrastructure on Arctic Operations". [DRDC-RDDC-2021-R024], Defence Research and Development Canada.

Jacobson, E., Argon, N., and Ziya, S. 2012. "Priority assignment in emergency response". *Operations Research,* 60:813-832.

Kahn, C. A., Schultz, C. H., Miller, K. T., and Anderson, C. L. 2009. "Does START Triage Work? An Outcomes Assessment After a Disaster". *Annals of Emergency Medicine,* 54(3):424-430.

Kwok, P. K., Yan, M., Chan, B. K., and Lau, H. 2019. "Crisis management training using discrete-event simulation and virtual reality techniques". *Computers & Industrial Engineering,* 135:711-722.

Lumbroso, D., and Davison, M. 2018. "Use of an agent-based model and Monte Carlo analysis to estimate the effectiveness of emergency management interventions to reduce loss of life during extreme floods". *Journal of Flood Risk Management,* 11(51):5419-5433.

Mills, F. A. 2016. "A simple yet effective decision support policy for mass-casualty triage". *European Journal of Operational Research,* 253(3):734-745.

Powell, W. B. 2022. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions.* [1st ed.] Hoboken, New Jersey: Wiley.

Rempel, M. 2024. "Modelling a major maritime disaster scenario using the universal modelling framework for sequential decisions". *Safety Science,* 171:106379.

Rempel, M., and Shiell, N. 2022. "Using Reinforcement Learning to Provide Decision Support in Multi-Domain Mass Evacuation Operations". *Operations Research and Analysis Conference.* Oct. 17th-18th, 2022, Copenhagen, Denmark.

Rempel, M., Shiell, N., and Tessier, K. 2021. "An approximate dynamic programming approach to tackling mass evacuation operations". *IEEE Symposium Series on Computational Intelligence*, 5th-7th Dec. 2021, Orlando, Florida, 1-8.

Schoenharl, T., and Madey, G. 2011. "Design and Implementation of An Agent-Based Simulation for Emergency Response and Crisis Management". *Journal of Algorithms & Computational Technology,* 5(4):601-622.

Wagner, N., and Agrawal, V. 2014. "An agent-based simulation system for concert venue crowd evacuation modeling in the presence of a fire disaster". *Expert Systems with Applications,* 41(6):2807-2815.

Wang, C., Ding, M., Yang, Y., Wei, T., and Dou, T. 2022. "Risk Assessment of Ship Navigation in the Northwest Passage: Historical and Projection". *Sustainability,* 14(9):5591.

Wang, S., Van Schyndel, M., Wainer, G., Rajus, V. S., and Woodbury, R. 2012. "DEVS-based Building Information Modeling and simulation for emergency evacuation". In *2012 Winter Simulation Conference (WSC)*, 1-12. https://doi.org/10.1109/WSC.2012.6465087.

Weber, B. (2016, Sep. 18). *Northerners consider new cruise ship rules after Crystal Serenity's voyage*. (The Canadian Press) https://www.cbc.ca/news/canada/north/new-rules-for-arctic-cruises-1.3767846

Zeigler, B. P., Muzy, A., and Kofman, E. 2019. *Theory of Modeling and Simulation*. [3rd ed.] London, United Kingdom: Academic Press.

Zhang, B., Chan, W. K., and Ukkusuri, S. V. 2014. "On the modelling of transportation evacuation: an agent-based discrete-event hybrid-space approach". *Journal of Simulation,* 8(4):259-270.

AUTHOR BIOGRAPHIES

HAZEL GRIFFITH is a Ph.D. student in the Department of Systems and Computer Engineering at Carleton University in Ottawa, Canada. She holds a M.A.Sc. in Electrical Engineering with concentration Modeling and Simulation from Carleton University. Her research interests include DEVS, machine learning, agent-based modelling, modelling human behavior with DEVS and agent-based modelling to study the spread of disease. Her email address is hazelgriffith@cmail.carleton.ca.

GABRIEL A. WAINER is Professor in the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). He is the head of the Advanced Real-Time Simulation lab, located at Carleton University's Centre for Advanced Simulation and Visualization (V-Sim). He is an ACM Distinguished Speaker and a Fellow of SCS. His email address is gwainer@sce.carleton.ca.